Name: Hung Hong

Title: Report for ProcJam Submission – Terrain Explorer

Link to Entry: https://hphong.itch.io/terrain-explorer

OBJECTIVE

As an entry of #procjam, an annual Procedural Content Generation (PCG) themed game jam, this project aims to meet the most important requirement, which is to "make something that makes something". Since procedural generation is a new concept that I have never applied before, my initial goal was to get familiarize with the concept and set myself a high goal of making a complicated RPG game that involves 6 inter-connected maps, similar to 6 sides within a box dimension. By setting a high goal, I can later scope my project accordingly to match with my current skill and knowledge. What I hoped to achieve was to make a really captivating and complex puzzle RPG game using the PCG concept using various techniques that I learned during class and from the reading assignments.

OUTCOME

Slowly as I worked on the project, I realized that my initial goal was overambitious and required a lot of knowledge and time to implement. As a result, I kept the initial idea of map design using PCG and discarded the idea of "inter-connected worlds" to adopt the plan of RPG-themed map generation. Inspired from the Pokemon series which has intrigue 2D map design with different entities, my end goal was changed to creating an RPG map generator in a Pokemon-inspired environment which supports different forms of terrain and aids in mapping and tiling. By generating well-designed maps that are ready to be used in game development, the application aims to ease the exhausting task of level design, especially for games with a huge number of maps, through PCG concept and make the development of games with similar genres much easier.

The product that I end up creating for this project is an application that generates various map designs and allows users to playtest the map by traversing the map with an in-game character. The application is made entirely from GameMaker Studio 1.4 and has 2D pixel setting similar to those in many Pokemon games using customized tilesets to suit the game dimension. The character that is used to discover the map is self-made by me and can only move around to discover the map in the perspective similar to an in-game perspective of a player, due to the fact that this is not an actual game and map design generation is prioritized over gameplay.

The application allows the user to switch between 2 perspectives: In-Game Perspective and Complete Perspective (as seen in Fig. 1). The user can toggle between the perspective by pressing the Spacebar. This is to give the game developer general setting of the map, as well as what it actually looks like when the map is used in-game. The map can be separated into 3 different entities: Wall, Floor, Lake, Grass, and Trees.

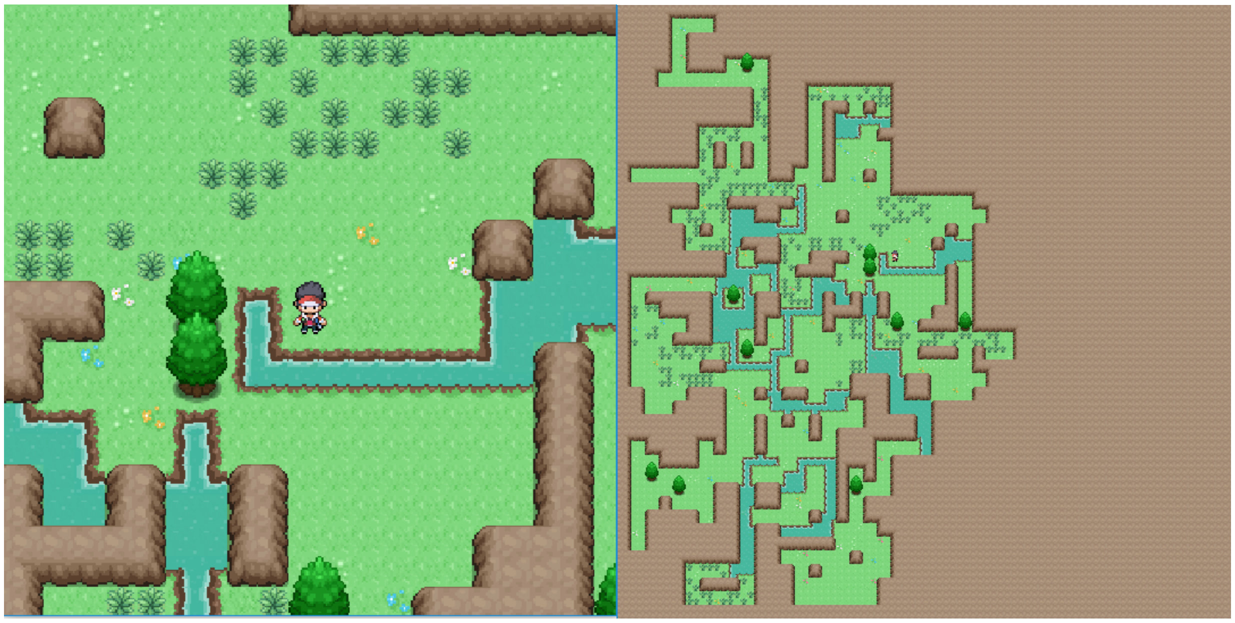

Fig 1. Side-by-side comparison between 2 perspectives of the application

The map is firstly defined by setting up Wall and Floor regions. Initially, the map is put into a grid system, and the controller is put in the middle of the map. This controller serves to define the Floor region by moving around the map in 1000 steps. Using the Random Walk technique, the Floor is defined by any place that the controller has moved to within the map. Afterwards, the Wall region is defined by wrapping around the Floor region. By using this technique, we can ensure that all parts of the Floor entity can be reached and reduce the risk of a stray entity. After the Floor and Wall regions are both set up, tiles are drawn accordingly using the Autotiling algorithm. Since the floor design is not uniform throughout as there are different patterns on the floor, the patterns are determined by Probability-based technique to make the floor looks more natural.

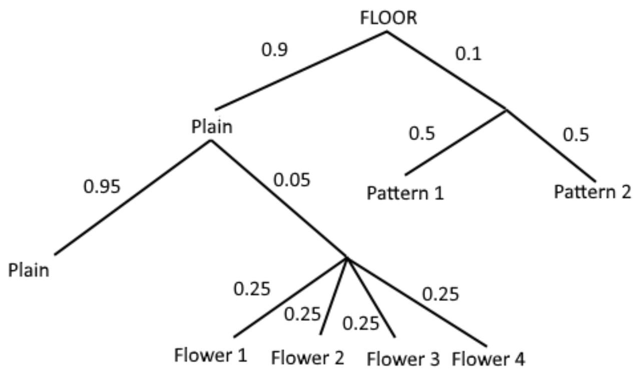

Fig 2. Probability for Floor Patterns

After the Floor and Wall regions are fully defined, the application traverses through the grid again to determine whether to place a controller to define Lake regions. After placing the controller successfully (with 1/20 chance of controller spawning), the Lake region is defined by the Random Walk technique similar to what I used for defining Floor region. The movement for the controller is confined within the Floor region, preventing it from intruding the Wall region. Afterwards, the application goes through the Autotiling algorithm, this time for the Lake region.

Trees are drawn on the map by using Probability-based Technique, with the concern of the surrounding region. For example, trees are more likely to spawn near to the lake. Trees are also more likely to spawn if it is covered by Wall regions since this means the placement of the tree will not interfere with the ability to access all areas on the map by the character. Grasses are defined by using Random Walk confined within the Floor region. However, the Autotiling process for the Grass region is Probability-based to vary the pattern of the grass covering the area.

Due to the nature of the application, the user can only be stopped by Wall region, meaning they can walk on Floor, Lake, and Grass fine. However, the regions for these areas are still separately defined for the convenience of future game development, e.g. random enemy spawning in the Grass region to enter battle (similar to Pokemon wild battle), or restricting entering the lake unless certain criteria are met.

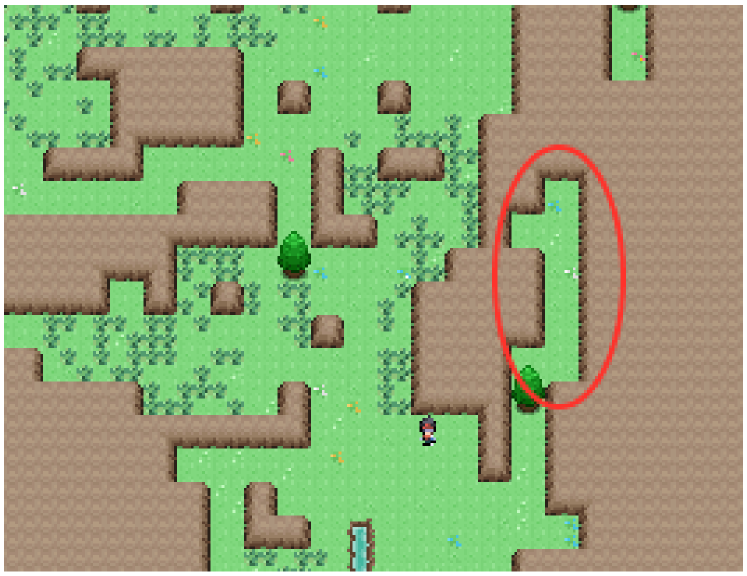
Fig 3. Weakness of Probability-based Tree Spawning

STRENGTHS AND WEAKNESSES

As discussed in the former section, the Random Walk technique is chosen to define a traversable region and prevent stray entities. The technique also allows for all areas on the map to be accessible and the overall design of the map to be natural since it reflects the exhaustive movements of the controller over the whole grid.

The Probability-based technique, which is really similar to the concept of Markov Chains, are used to give the map the overall "wilderness" and "natural" aspect. Region-based Probability makes sense with reality, such as trees are more likely to grow alongside water areas, and the combination with the Random Walk technique gives the map the "randomness" that it needs to make the map less systematic and more interesting to discover.

Current weaknesses of the Probability-based Method, especially when using for tree spawning, is the very low chance of blocking an area from being accessible. In some rare cases, it can hinder the travel path of the character completely since the location spawning for the character is fixed (in the middle of the grid). One way to prevent this from happening is to carry out a pathfinding map to ensure connectivity between areas, which I cannot carry it out in time for the submission of the project due to time constraint and difficulty.

REFLECTION

From making the project

I learned a lot from making the project on my own as I have not been using GameMaker engine recently and the project serves to remind me of how to utilize the engine to build this application. The project also allows me to learn about different techniques that can be used with the PCG concepts to generate well-designed maps without having to manually design it tile by tile. One part that takes a huge amount of my time was to implement the Autotiling algorithm, and by the end of the project, I learn a lot from my personal trials and errors.

From participating in the ProcJam community

This is my first game jam that I have ever participated and I have a great time learning from past experiences of people in the same field. I realized that I learned better by getting inspired by other people' works, learn from their successes and failures, and apply on my own experience for this jam. I am grateful to have the opportunity to get exposed to such event, and I am glad to participate in future events in the Game Development community to hone my skill and showcase my talent.